Bioinformatics, 2020
24 April 2020

Prediction of Protein Secondary Structure with fully connected Dense Neural Network

Ramya Kondapi[1, *]

[1]Master of Science, Computer Science, College of Engineering and Computer Science, University of Central Florida, Orlando, FL, 32816

*To whom correspondence should be addressed.

Abstract

Motivation: To determine the function of the protein, protein secondary structures plays a significant role. Furthermore, they also help in finding the drug design and to treat several diseases which means that it is important to determine the secondary structure. Since this is an expensive and challenging task to perform in a laboratory, a computational method to determine the secondary structure has been implemented. The method used in this paper is fully connected Dense Neural Network. The protein data used in the method is label encoded and trained using Keras Classifier, predicted and classified using Dense network while optimizing the error using Adam optimizer calculated through Categorical cross-entropy.

Results: The results obtained in this method are better when compared with the referred[1] paper studied to Predict the secondary structure of the same proteins used in the classification process in the proposed method. The best results obtained in this study are achieved at only 5 iterations 6 hidden layers, and 200 epochs, whereas the paper referred to this project achieved their best results at 8 iterations, 2 hidden layers, 200 epochs.

Availability: The code is available in this[2] GitHub link. The data can be collected from the Protein Data Bank (Commonly referred as PDB).

Contact: rkondapi@knights.ucf.edu

1 Introduction and Background

The function of the proteins is identified by the three dimensional protein configuration and a few particular characteristics of chemical compounds in amino acids of different living organisms like the homo sapiens and other animals, also in fungi, algae, and other microorganisms, also, in plants and these proteins help these organisms in several important intracellular and extracellular activities. There are four different structures where the proteins can be found, namely, primary structures, secondary structures, tertiary structures and quaternary structures. The primary structures is where the Amino acids constitutes by peptide bonds are present, the secondary structures is where the most frequently appeared secondary structures are present such as, α-helices, β-sheets, and coils, the tertiary structure is where the three dimensional conformation of all the amino acids is present, the tertiary structure is also called the integral structure of polypeptide chain, and finally the quaternary structure is where one or more polypeptide chains forming a functional protein is present.

This significance of proteins is annulled when the functionality of proteins deviates from their normal behavior. These abnormal functionalities of proteins, which are the misfolding of proteins caused by factors such as genetic mutations, aging, and other external elements can disastrously impact health of several people, particularly, the elderly resulting in several diseases such as lowering dopamine levels causing Parkinson's, senile dementia also called, Alzheimer's, and Type-2 diabetes. To lower the risk of this impact, the misfolded proteins are to be determined through the three-dimensional structure of the proteins present in primary structure i.e., the amino acid sequence. So, as it clearly indicates, the function of the protein is determined by the three-dimensional structure of the protein, and the three-dimensional structure of the protein is determined by the protein sequence. Although we can determine the three-dimensional structure of the protein the laboratory suing the methods such as nuclear magnetic resonance, X-ray diffraction, electron crystallography, it is considered an expensive and complicated process. Hence, a simpler and less-expensive method is by implementing computational models.

In this study, a fully functional dense neural network model has been proposed. While the other methods used in the original paper are efficient, this method proves that the method used in the original paper, Clonal Selection Algorithm for classification with Multilayer Perceptron, are not as efficient as the Dense Neural Network model. The accuracy achieved by this model are efficient and expeditious than the accuracy achieved through Clonal Selection Algorithm.

In the second section, the protein data used in this study is described. The third section has the information about the methods implemented in this study and the original study, including the results of each of these methods and their comparisons. Finally, the conclusion and insights.

2 Protein Data

The data used in this study is obtained from the Protein data bank, commonly known as PDB, this comprises of 3336 amino acid sequences of 22 different hemoglobin proteins. Each protein has 20 different amino acids in its primary structure, namely, A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y. These are all English alphabets apart from B, J, O, U, X, Z. There are different methods to assign a secondary structure to each of the amino acids in the protein, but the method used in this study to determine the secondary structure is through the Dictionary of Protein Secondary Structure (DSSP)[3]. Eight different secondary structures are used to assign, namely, H, G, I, E, B, T, S, and C. H is called the 4-turn α-helix (4 residues), G is the 3-turn α-helix (3 residues), and I is the 5-turn π-helix (5 residues). E is the extended parallel or anti-parallel β-sheet (2 residues). B is a single pair β-sheet or β-bridge hydrogen bond formation. T is the hydrogen bonded turn (3,4,5 turns). S is a bend, assigned to a non-hydrogen bond. And finally, C is a coil which is codded as a space or coil or dash or which are neither of the above conformations. The 22 protein data used is this study, 1a01, 1aj9, 1bz1, 1pzz, 1dxt, 1dxu, 1dxv, 1g9v, 1gli, 1hba, 1hbb, 1i3d, 1i3e, 1ljw, 1mko, 1o1j, 1o1l, 1o1p, 1qsh, 1y0t, 1y0w, and 1yzi, are downloaded from PDB[4] in .pdb format. All the amino acids in these .pdb files are assigned to a protein secondary structure by uploading each of the files in the DSSP[3]. The final data to work in this study is present in all the .dssp files obtained from DSSP. Since, working with 8 different classes (8 different secondary structures) is fairly a difficult task, there are few reductions applied such as converting the secondary structures {H, G} to {H}, {E, B} to {E}, rest to {C}. This leaves us only 3 classes to work on.

There are a total of 29 attributes in the dataset: '#', 'RESIDUE', 'AA', 'STRUCTURE', 'BP1', 'BP2', 'ACC', 'N-H-->O1', 'O-->H-N1', 'N-H-->O2', 'O-->H-N2', 'TCO', 'KAPPA', 'ALPHA', 'PHI', 'PSI', 'X-CA', 'Y-CA', 'Z-CA', 'CHAIN', 'AUTHCHAIN', 'NUMBER', 'RESNUM', 'BP1', 'BP2', 'N-H-->O', 'O-->H-N', 'N-H-->O', 'O-->H-N','ProteinID'. The last column is appended at the end as a unique value representing each protein, making a total of 30 attributes. Few of these features are dropped as they seemingly do not contribute to the results obtained by the model. These attributes are either duplicates or null values. Whereas a new feature is introduced to the dataset called "Label" this contains all the three classes H, E and C.

3 Proposed method and original paper method

There are several stages during the implementation of the model, starts with preprocessing of the data, which also includes vectorization of the data followed by training the data obtained, next is to initialize the network with neurons, activation functions, number of hidden layers, and number of epochs, followed by initializing a method to calculate the error, and also a method to optimize it further, and finally using a validation technique to compute the accuracy, showed in Fig 1.

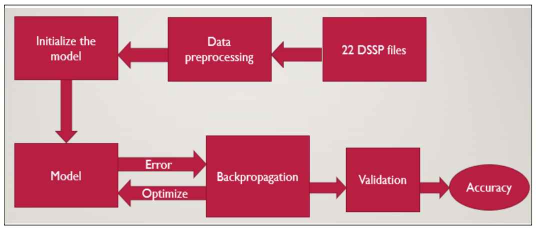

Fig.1. Model Pipeline

The first stage starts with preprocessing the data where all the irrelevant features are omitted, also new features are introduced to the dataset such as ProteinID and Label. In this stage, all the categorical data are converted to numerical data through a method called label encoding. In this type of encoding, all the unique values in a column are considered and assigned a unique integer value to each of the unique values. For instance, consider a column with unique values ['A', 'B', 'C'], each of these categorical values are assigned an integer values: A → 0, B → 1, C →2. This can be done using a predefined method LabelEncoder() from the sklearn library.

One the data is preprocessed, all the required elements in the neural network are initialized. In the current model a count of 100 neurons have been initialized, and there are several numbers of hidden layers 1 through 6, and several number of varying epochs [10,50,80,200] are initialized and implemented the model with combination of each hidden layer and each number of epochs. The purpose of a neuron is, as the name suggests, it is similar to the neural architecture in a human brain. Its basic functionality is that it takes input and produces an output based on an activation function chose In this model the activation function applied to the neuron is a common activation function used in a deep learning model, rectified linear unit given by the equation below:

$$f(x) = max(0, x)$$

There are several activation function to use, such as sigmoid, tanh, step function etc. But compared to these models ReLU is found to be perform better by taking the value 0 as an output when the input is negative, and when the input value is a positive integer, it returns the input itself. ReLU activation function is used for both input layer and hidden layer, whereas another activation function called softmax is applied to the final output layer. The softmax function works in this way: it maps each output to a range of [0,1], and also makes sure that the sum of all the outputs is equals to 1, making the output a probability distribution. Each hidden layer is a mathematical function designed to produce an output specific to the given input. A hidden layer is basically a non-linear transformation of the inputs entered into the network. Each hidden layer identifies a specific characteristic of the given input, hence the more the hidden layer, the better the inputs gets classified. Our model has varying hidden layers from 1 to 6. An epoch provides more insight to data, meaning, it provides better generalization of the data by cycling throughout the training set. Each epoch refers to one cycle throughout the training set.

After initializing the model, we will pass the inputs to the model that we have initialized. The model takes the input processes through several hidden layers and finally the output layer provides the output. After generating the output, error will be calculated using categorical cross entropy. Now it is the time to optimize the error that was calculated, this will be done using Adam optimizer which is the combination of Root mean square prop and stochastic gradient descent with momentum. This optimization is done by going back to all the hidden layers from the output layer and readjusting the weights. Once the weights are updated, the output is generated again by the same process as above, this is called backpropagation. Once the outputs are finalized, the accuracy is achieved by applying tenfold cross validation technique which simply means that predictive models are evaluated by dividing the training set into different sets to train the model, and evaluate it using the test set. Below is a sample neural network of how the model actually looks like.

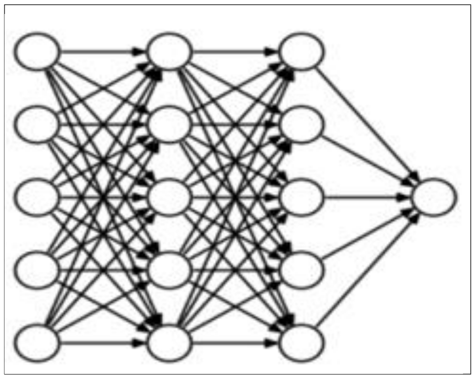

Fig. 2: Neural network model

Here the left extreme layer is called the input layer, and the extreme right layer is called the output layer with 2 hidden layers in between. There can be many numbers of hidden layers as mentioned above but more the hidden layers the better.

Let us compare this proposed model with the model proposed by the original paper. The model they implemented is a two-phase method, first protein data were improved with Clonal Selection algorithm, and then the data trained by CSA, the untrained data and the independent data are sent to the multi-layer perceptron to classify the data. The model execution was done with the same epochs and same layers that I considered. CSA is one of the most studied algorithm of Artificial Immune System, its main principle is to provide the antibody diversity that can fight against the antigens. CSA is a Class of algorithms that explains how B and T lymphocytes improve affinity maturation. There are three techniques: CLONALG, AIRS, BCA. The technique used by the original paper is CLONALG where there are 2 sets of cells: a set of memory cells, a set of non-memory cells. Real-valued vectors are used instead of commonly used bit-strings to represent cells. More than one memory cell used per class. We have three different classes: H, E, C and trained multiple MLPs in parallel for each class. Basically, each MLP denotes to the recognition of a different class of the input data. Below is a flow chart of their proposed model.

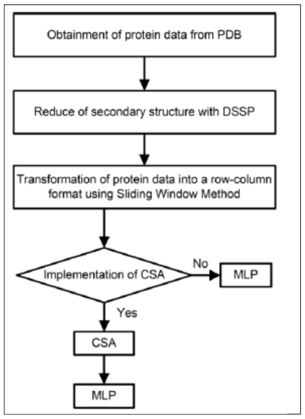
Obtainment of protein data from PDB
Reduce of secondary structure with DSSP
Transformation of protein data into a row-column format using Sliding Window Method
Implementation of CSA
No
MLP
Yes
CSA
MLP

Fig. 3: Flow chart of the model in the original paper

A multilayer perceptron is a feedforward neural network. Each hidden layer identifies a specific characteristic of the given input, hence the more the hidden layer, the better the inputs gets classified. Our model has varying hidden layers from 1 to 6. An epoch provides more insight to data, meaning, it provides better generalization of the data by cycling throughout the training set. Each epoch refers to one cycle throughout the training set, we will pass the inputs to the model that we have initialized. The model takes the input processes through several hidden layers and finally the output layer provides the output. The difference between the MLP and Dense neural network is that in dense neural network each neuron in the hidden layer receives input from all the neurons from the previous layer, whereas in their study they have used a dropout, which means that using a predefined Dropout() we pass an argument of how much percentage of neurons are supposed to be dropped, then the number of neurons are selected at random are set to 0, which means that those neurons do not contribute to the output. This might be disadvantageous when the unset neurons contain crucial information in generating the output. The dropout is performed to avoid overfitting of the neurons in the model, basically it means to simplify the neural network. Below is sample neural network how their model looks like.

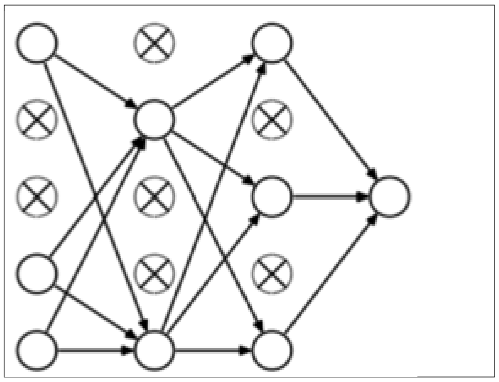

Fig.3: MLP with dropout

4 Results from the method implemented and the original paper

The results obtained using the implemented model are below:

Epoch	Hidden Layers					
	1	2	3	4	5	6
10	92.652	93.28	93.65	93.92	94.43	94.59
50	94.37	95.47	95.67	96.23	96.46	96.54
80	94.74	95.73	96.17	96.40	96.44	96.24
200	94.92	95.69	96.468	96.685	94.282	93.85

Table 1. Outputs from the proposed model for 5 iterations

As seen in the table above, the maximum accuracy is achieved for 5 iterations is at 4 hidden layers, and 200 epochs. Whereas below are the outputs from the original paper:

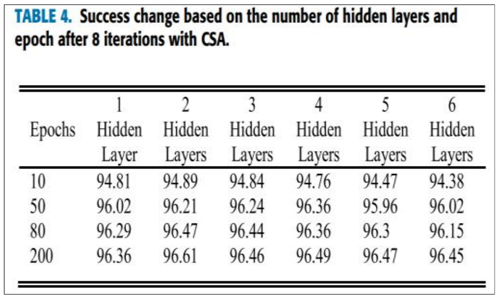

TABLE 4. Success change based on the number of hidden layers and epoch after 8 iterations with CSA.

Epochs	1 Hidden Layer	2 Hidden Layers	3 Hidden Layers	4 Hidden Layers	5 Hidden Layers	6 Hidden Layers
10	94.81	94.89	94.84	94.76	94.47	94.38
50	96.02	96.21	96.24	96.36	95.96	96.02
80	96.29	96.47	96.44	96.36	96.3	96.15
200	96.36	96.61	96.46	96.49	96.47	96.45

Fig.4: Results from the original paper

As seen in the figure above, the original paper achieved their best results for 8 iterations, for 2 hidden layers and 200 epochs. This maybe due to reason of applying dropout.

Conclusion

By this we can say that Better accuracy can be achieved with more neurons sooner in less number of iterations without applying dropout. If a dropout has to be applied in order to avoid overfitting then consider the neurons that contribute less to the outputs instead of choosing them randomly.

Insights

This work is an improvement of the original paper[1]. We are currently considering 100 neurons per layer, further, we need to understand how many neurons required for achieving better accuracy sooner than later. Furthermore, combining LSTM and Dense neural network may improve accuracy. Since LSTM vanishing gradient problem, accuracy may increase with GRU and Dense neural network.

References

Burcu Çarkli Yavuz, Nilüfer Yurtay, Ozhan Ozkan. “Prediction of Protein Secondary Structure With Clonal Selection Algorithm and Multilayer Perceptron” August 2018.

A. Lanaridis, V. Karakasis, A. Stafylopatis. “Clonal Selection-based Neural Classifier” Sept 2008

https://github.com/sam1918/BioProjectCode.git

https://www.i2tutorials.com/deep-learning-interview-questions-and-answers/what-do-you-mean-by-dense-layer-and-drop-out-layer-in-keras-neural-network/

J. Zhou and O. G. Troyanskaya, ``Deep supervised and convolutional generative stochastic network for protein secondary structure prediction,'' in Proc. 31st Int. Conf. Mach. Learn., vol. 32, 2014, pp. 745 753.

O. Ozkan et al., ``A study on the effects of sympathetic skin response parameters in diagnosis of bromyalgia using arti cial neural networks,'' J. Med. Syst., vol. 40, no. 3, p. 54, Mar. 2016.

B. Ç. Yavuz, N. Yurtay, and O. Ozkan, ``Estimation of secondary structure of hemoglobin protein by multilayer feeding arti cial neural aggregates,'' in Proc. 3rd Int. Congr. Eng., Archit. Design, Kocaeli, Turkey, 2018, pp. 65 66.

S.Wang, J. Peng, J. Ma, and J. Xu, ``Protein secondary structure prediction using deep convolutional neural elds,'' Sci. Rep., vol. 6, Jan. 2016, Art. no. 18962.

H. Hasic, E. Buza, and A. Akagic, ``A hybrid method for prediction of protein secondary structure based on multiple arti cial neural networks,'' in Proc. 40th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO), May 2017, pp. 1195 1200.

N. Qian and T. J. Sejnowski, ``Predicting the secondary structure of globular proteins using neural network models,'' J. Mol. Biol., vol. 202, no. 4, pp. 865 884, Aug. 1988.

L. H. Holley and M. Karplus, ``Protein secondary structure prediction with a neural network,'' Proc. Nat. Acad. Sci. USA, vol. 86, no. 1, pp. 152 156, Jan. 1989.

L. H. Holley and M. Karplus, ``Protein secondary structure prediction with a neural network,'' Proc. Nat. Acad. Sci. USA, vol. 86, no. 1, pp. 152 156, Jan. 1989.